Implementation of an Important Wave Model on Parallel Architectures

Tim Campbell

Mississippi State University

John Cazes

Northrop Grumman IT

Erick Rogers

Naval Research Laboratory

Outline

- Overview of SWAN Model
  - Model Formulation
  - Numerical Implementation
  - Software Design
- Parallel Implementation
- Parallel Performance
- Summary

Model Formulation

- SWAN -- Simulating WAves Nearshore
- Third-generation numerical wave model to compute realistic estimates of wave parameters in coastal regions, lakes and estuaries from given wind, bottom, and current conditions
- Based on wave action balance equation:
  - Wave action density spectrum: $N(x,y,\sigma,\theta;t)$
  - Source and sink terms $S(\sigma,\theta)$
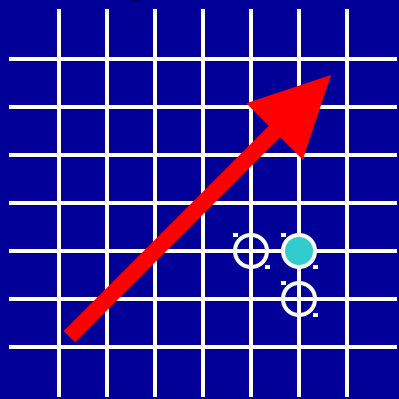
$$\frac{\partial}{\partial t}N + \frac{\partial}{\partial x}c_x N + \frac{\partial}{\partial y}c_y N + \frac{\partial}{\partial \sigma}c_\sigma N + \frac{\partial}{\partial \theta}c_\theta N = \frac{S}{\sigma}$$
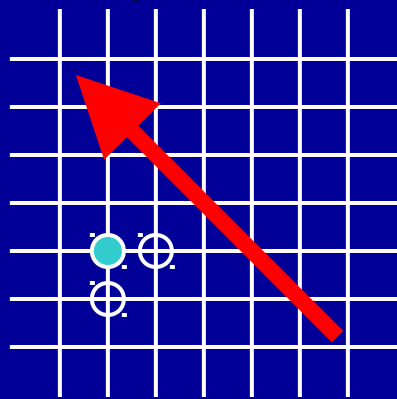
Numerical Implementation

- Finite difference schemes in all five dimensions (time, geographic, and spectral space)
- Geographic space: implicit second-order upwind scheme
- Spectral space: implicit first-order upwind scheme, supplemented with a central approximation
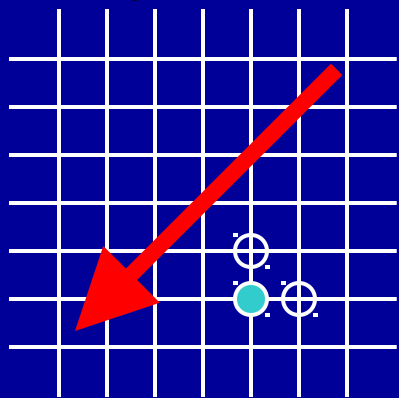
Numerical Implementation (cont.)
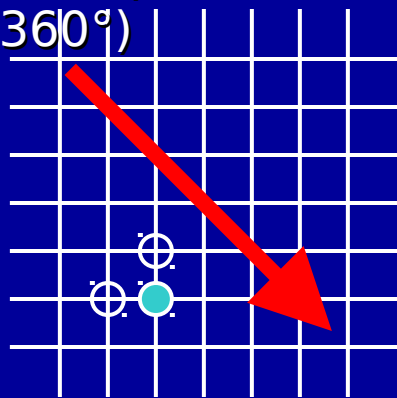
Sweep 1 (0-90°)

Sweep 2 (90-180°)

Sweep 3 (180-270°)

Sweep 4 (270-360°)

- Spectral space decomposed into 4 quadrants
- Iterative sweep over spatial grid to account for interactions between spectral quadrants:
  - Refraction
  - frequency shifting
  - nonlinear source terms
- Matrix solution at each geographic grid point:
  - Tri-diagonal: no currents and stationary depth
  - Banded: currents or non-stationary depth

Software Design: Main Program

```
    SUBROUTINE SWMAIN
    …process input…
    …setup "POOL" array…
    DO IT = 1, MTC
        …update BCs…
        …update input fields…
        CALL SWCOMP
        …process output…
    END DO
    …finalize…
    END SUBROUTINE SWMAIN
```

Software Design:
# Main Computational Subroutine

```
        SUBROUTINE SWCOMP
        DO ITER = 1, ITERMX
            DO SWPDIR = 1, 4
                …set sweep parameters…
                DO IY = IY1, IY2, IDY
                    DO IX = IX1, IX2, IDX
                        CALL SWOMPU(…,IX,IY,…)
                    END DO
                END DO
            END DO
            …check convergence…
        END DO
        END SUBROUTINE SWCOMP
```

Outline

- Overview of SWAN Model
- Parallel Implementation
  - Dependency & Concurrency Analysis
  - Pipelined Parallel Approach
  - Software Design Using OpenMP
  - Theoretical Scaling
- Parallel Performance
- Summary

Dependency & Concurrency
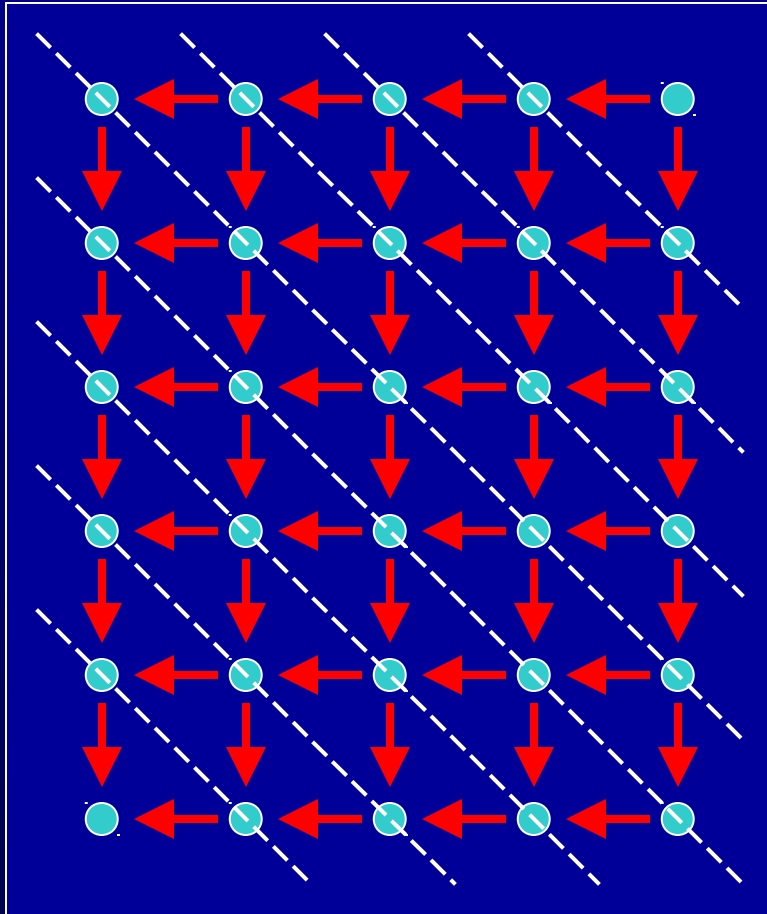
- Horizontal and vertical arrows indicate data dependencies
- Diagonal dashed lines connect points with no dependencies among them that can be computed in parallel

Pipelined Parallel Approach

Threads

3 = (yellow)

2 = (blue)

1 = (green)

0 = (red)
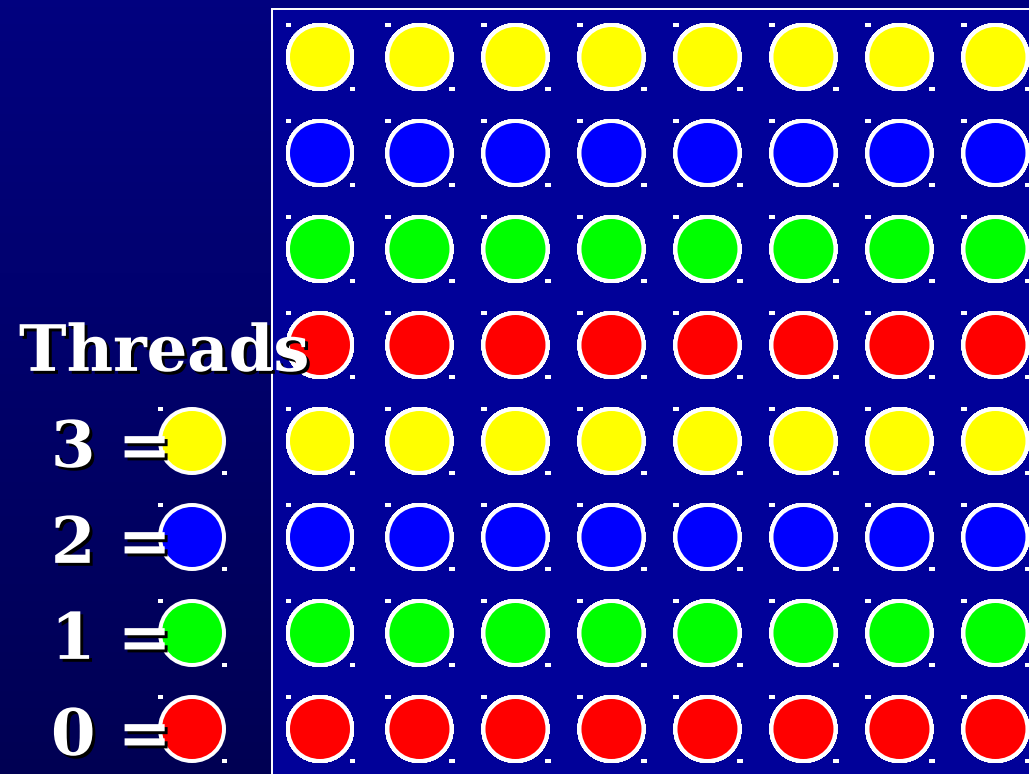
- Example on 8x8 Grid With 4 Threads
- Each thread assigned a row in round-robin fashon
- Parallel computations occur on points with no dependencies among them

Software Design Using OpenMP

```fortran
!$OMP PARALLEL DEFAULT(SHARED)
!$OMP+PRIVATE(ITER,SWPDIR,…)
      DO ITER = 1, ITERMX
        DO SWPDIR = 1, 4
          …set LLOCK array…
          …set sweep parameters…
!$OMP DO SCHEDULE(STATIC,1)
          DO IY = IY1, IY2, -IDY
            DO IX = IX1, IX2, -IDX
              DO WHILE(LLOCK(IX,IY+IDY)
!$OMP FLUSH

              END DO
              CALL SWOMPU(…,IX,IY,…)
              LLOCK(IX,IY) = .FALSE.
            END DO
          END DO
        END DO
        …check convergence…
      END DO
!$OMP END PARALLEL
```

- Large parallel region over whole subroutine
- Pipelined parallel on IY loop using OMP DO
- Data dependencies checked using LLOCK: TRUE (i.e., locked) until computation finished at grid point
- Details not shown:
  - Barriers
  - Serial regions
  - Work array (de)allocation

Theoretical Scaling of Pipelined Parallel Approach
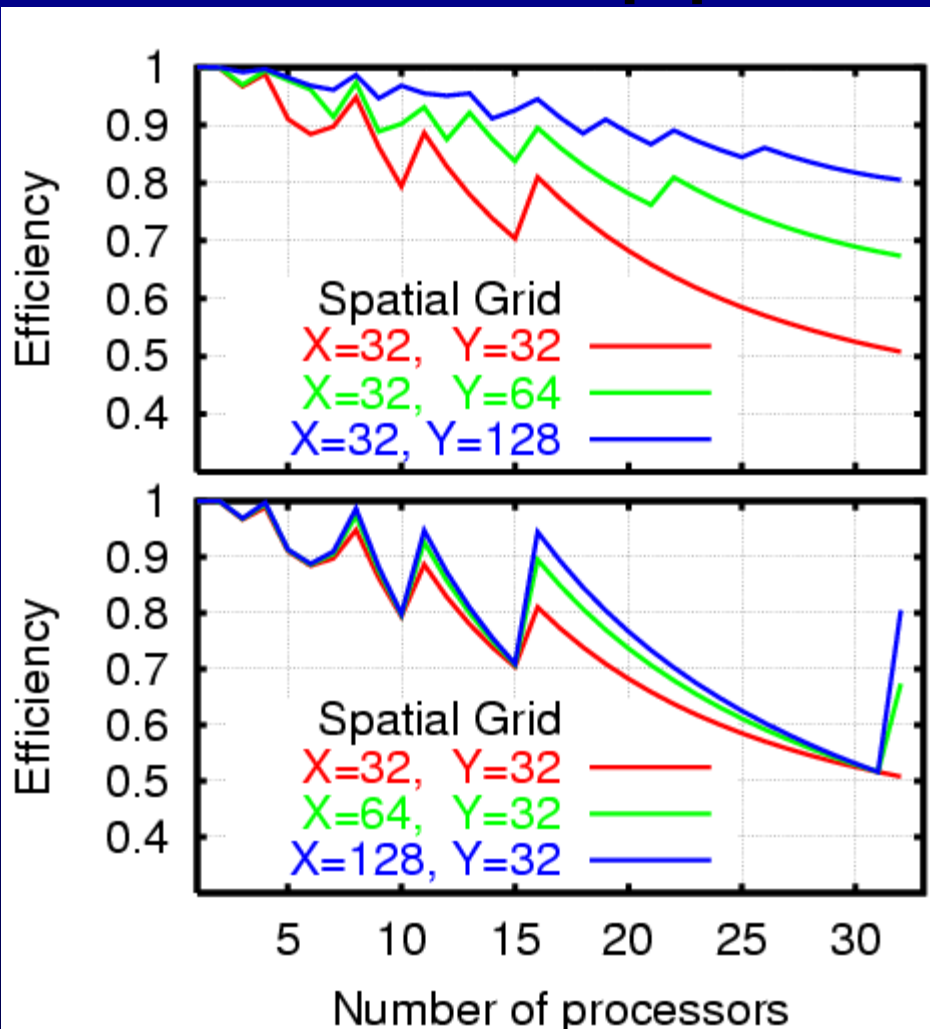
$$S_{ideal}\left(X,Y;P\right)=\frac{XY}{X\left\lceil\frac{Y}{P}\right\rceil+\left(Y-1\right)\bmod P}$$

$$E_{ideal}\left(X,Y;P\right)=\frac{S_{ideal}\left(X,Y;P\right)}{P}$$

- *P* ≤ min (*X,Y*)
- Peaks where ceil(*Y/P*) decreases
- For best performance: *P* ≤ min(*X,Y*)/2

Outline

- Overview of SWAN Model
- Parallel Implementation
- Parallel Performance
  - Comparison With Ideal
  - Performance on Different Platforms
  - Changing Workload Per Grid Point
  - Problem Constrained Scaling
  - Memory Constrained Scaling
  - Case Examples
- Summary

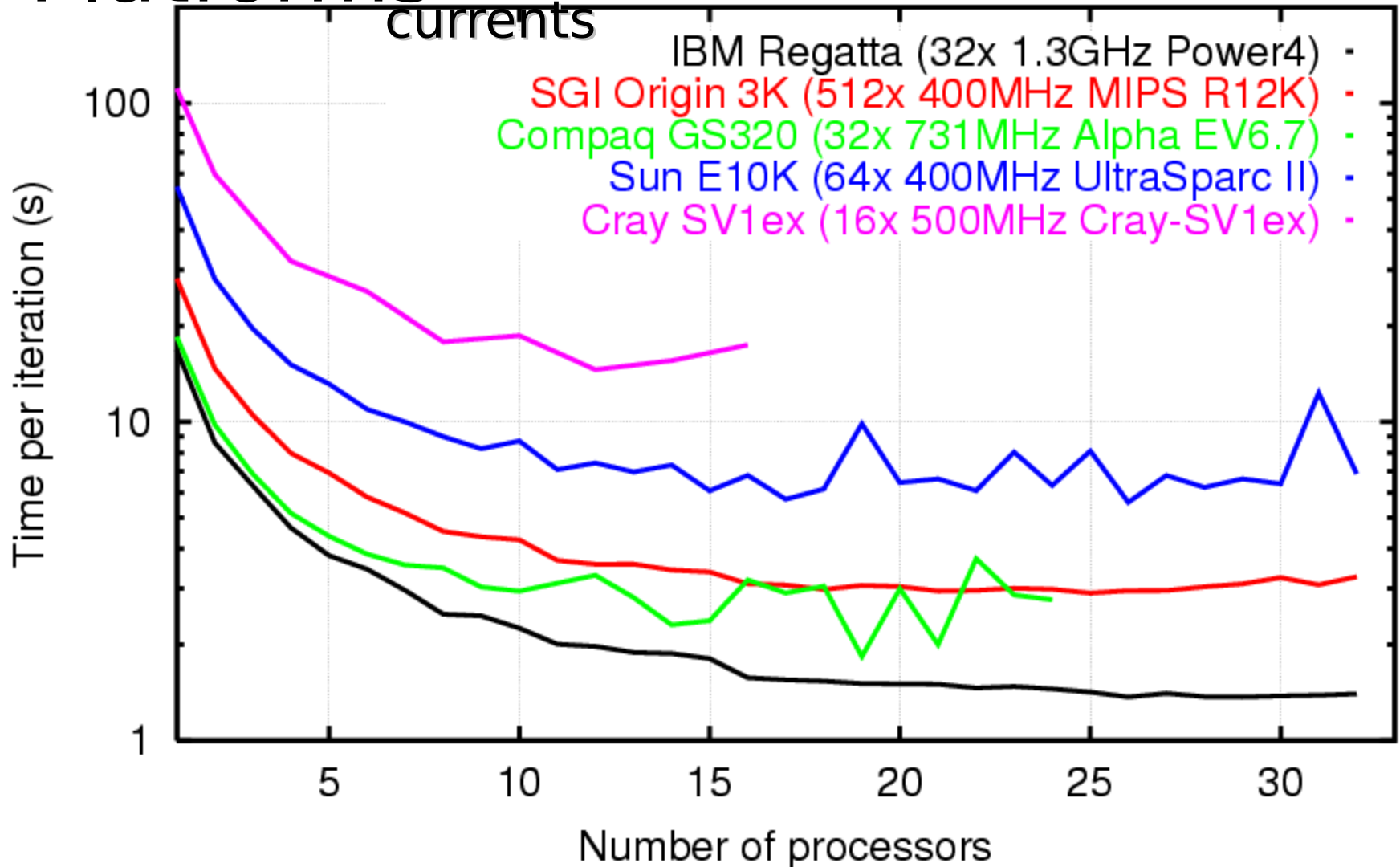# Actual Performance: Comparison With Ideal
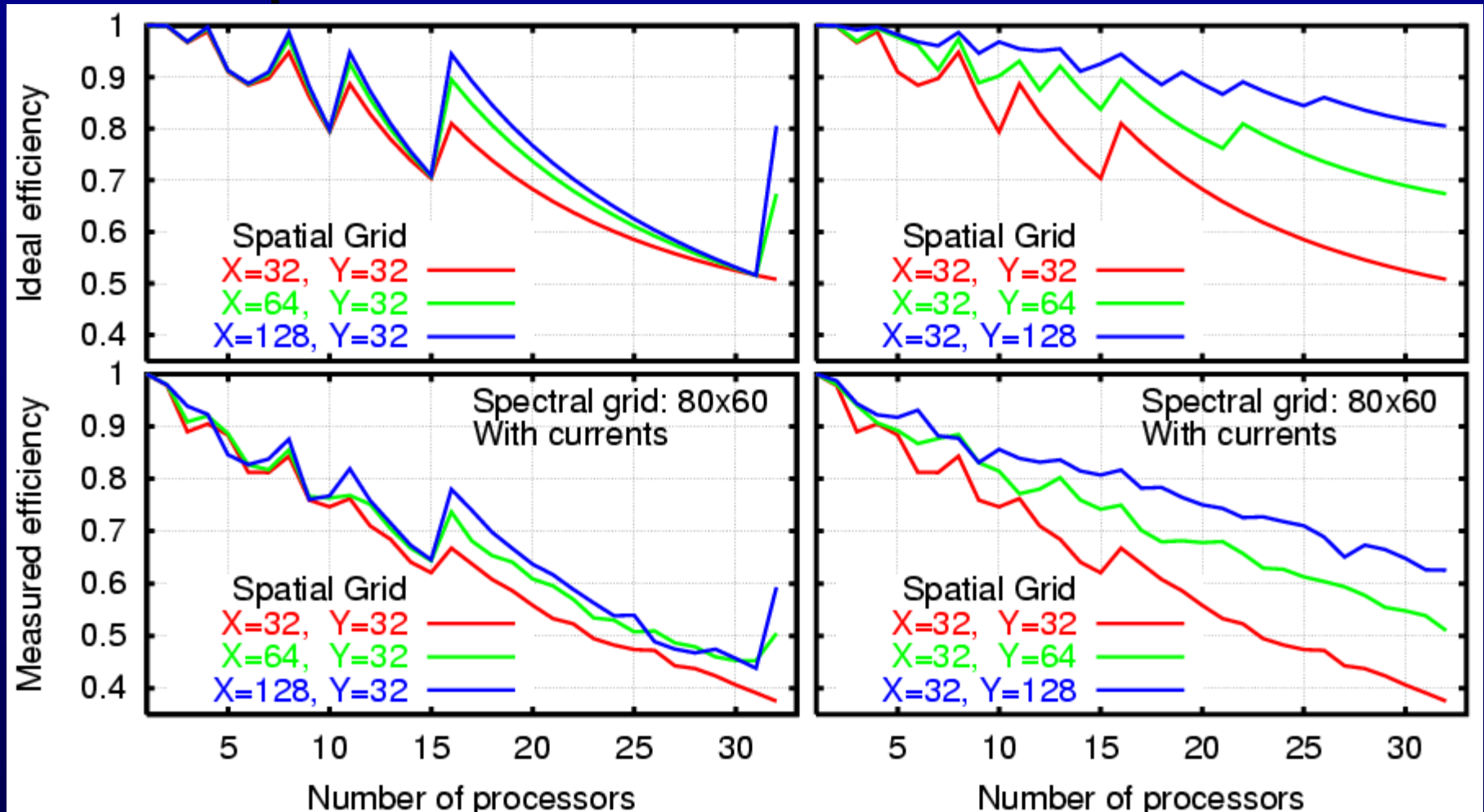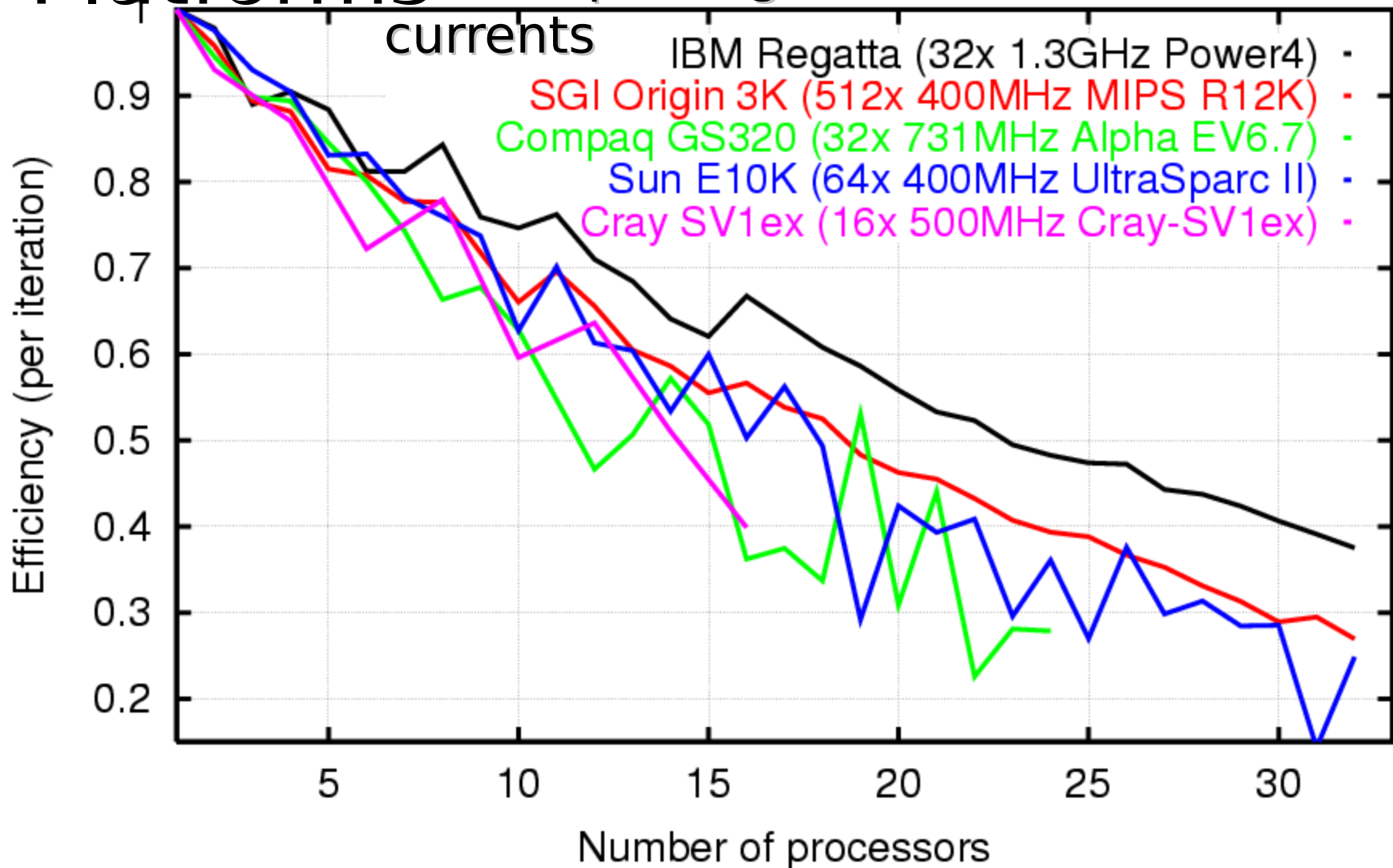

IBM Regatta (32x 1.3GHz Power4)

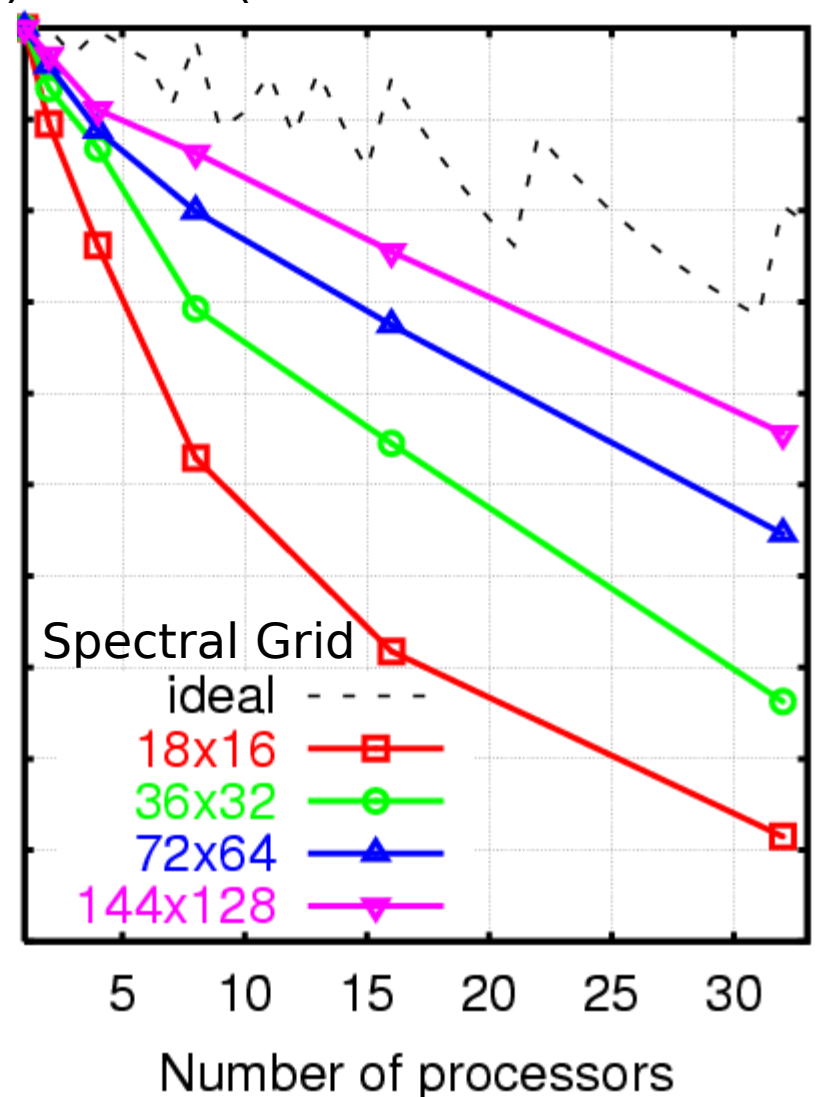# Performance on Different Platforms

32x32 spatial grid with no currents

IBM Regatta (32x 1.3GHz Power4)
SGI Origin 3K (512x 400MHz MIPS R12K)
Compaq GS320 (32x 731MHz Alpha EV6.7)
Sun E10K (64x 400MHz UltraSparc II)
Cray SV1ex (16x 500MHz Cray-SV1ex)

Time per iteration (s)

100

10

1

5   10   15   20   25   30

Number of processors

Efficiency on Different Platforms

32x32 spatial grid with no currents

Efficiency (per iteration) vs Number of processors

IBM Regatta (32x 1.3GHz Power4)
SGI Origin 3K (512x 400MHz MIPS R12K)
Compaq GS320 (32x 731MHz Alpha EV6.7)
Sun E10K (64x 400MHz UltraSparc II)
Cray SV1ex (16x 500MHz Cray-SV1ex)

Changing Workload Per Grid

IBM Regatta (32x 1.3GHz Power4) SGI O3K (512x 400MHz MIPS R12K)

64x64 spatial grid with no currents

Spectral Grid
- ideal — — —
- 18x16
- 36x32
- 72x64
- 144x128

Efficiency (per iteration)

Number of processors
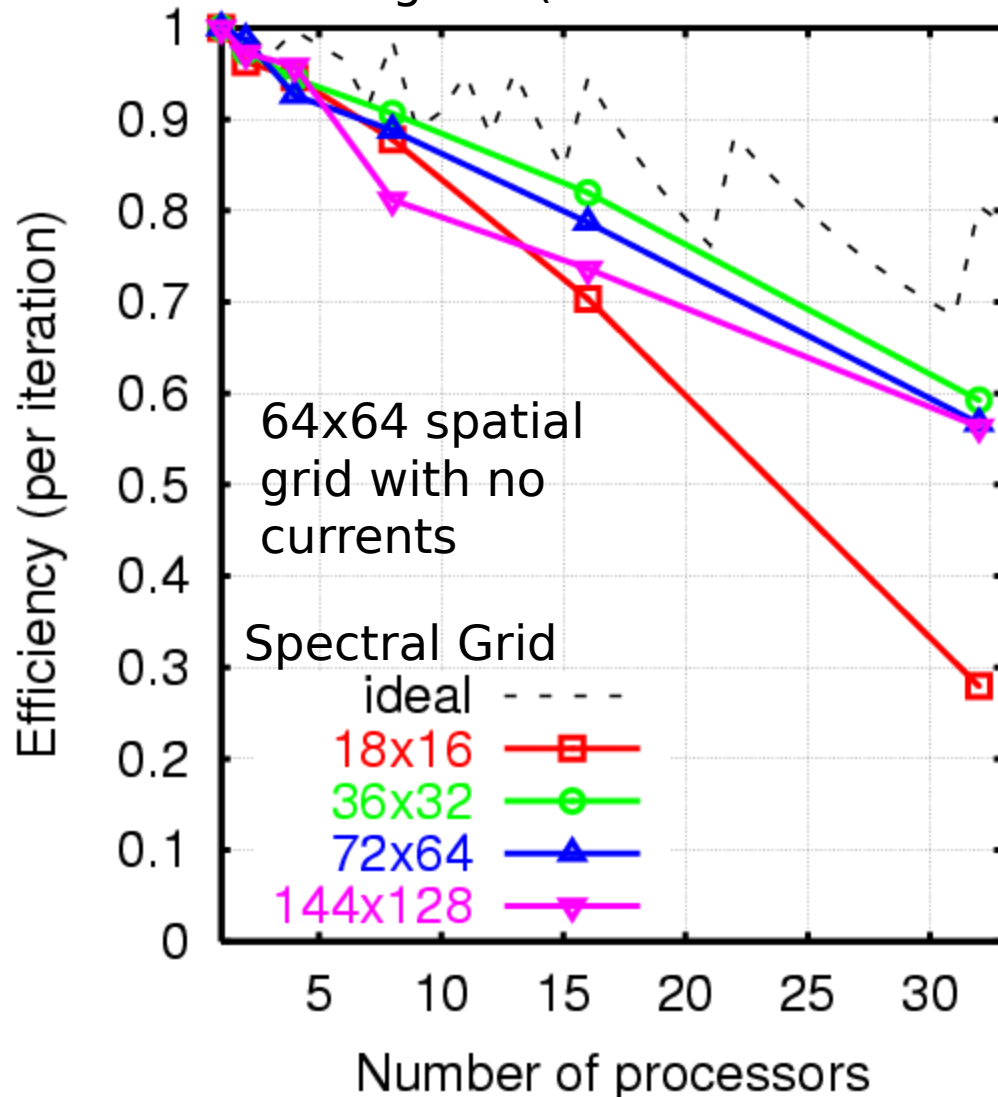
Problem Constrained Scaling

IBM Regatta (32x 1.3GHz Power4)SGI O3K (512x 400MHz MIPS R12K)

Efficiency (per iteration)

36x32 spectral grid with no currents

Spatial Grid
1K
4K
16K
64K
256K

Number of processors

Memory Constrained Scaling - IBM

36x32 spectral grid with no currents

Spatial Grid
P*32 K
P*64 K
P*128 K

Efficiency (per iteration)
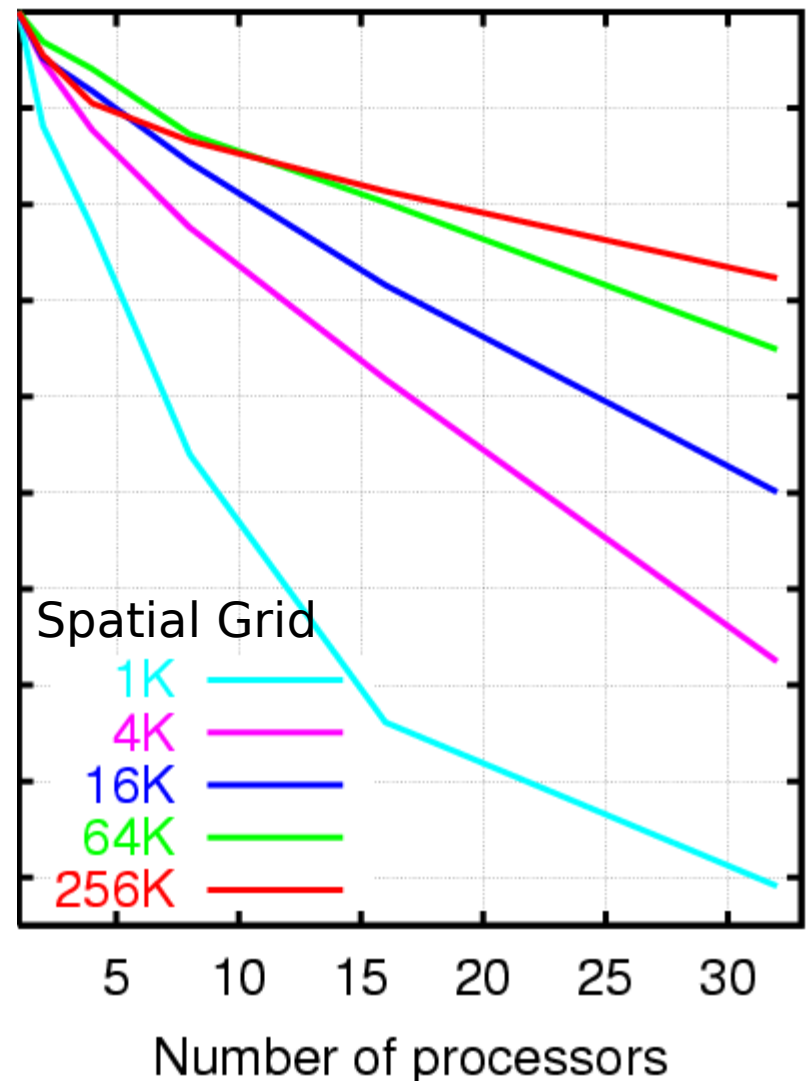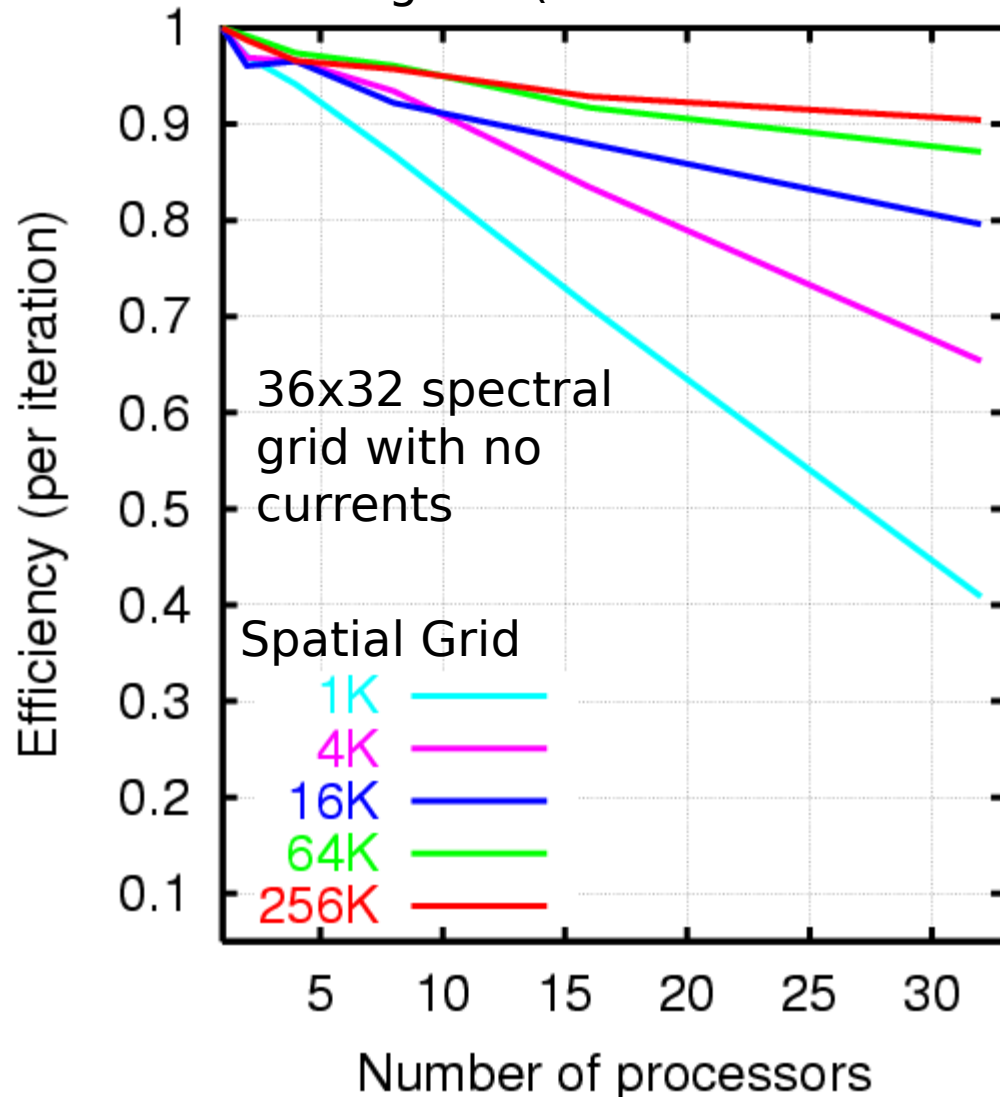
Number of processors (P)
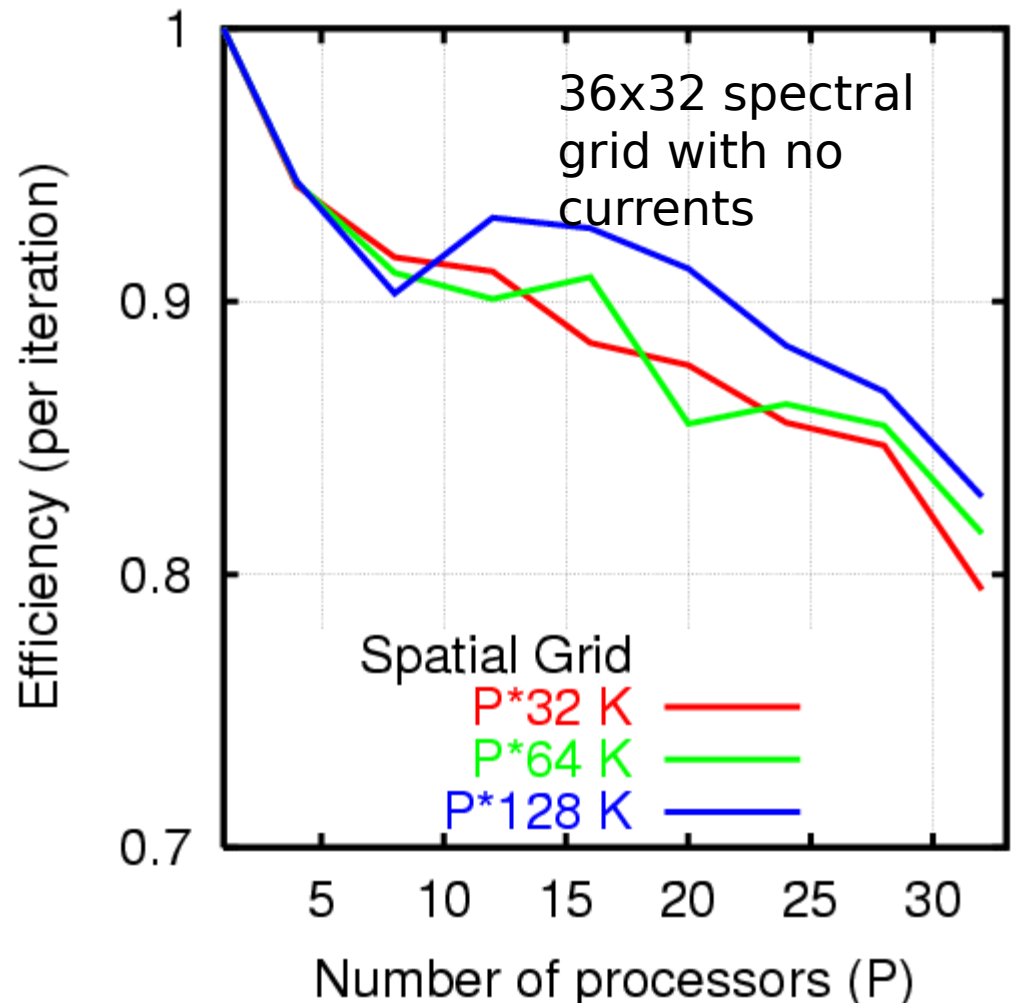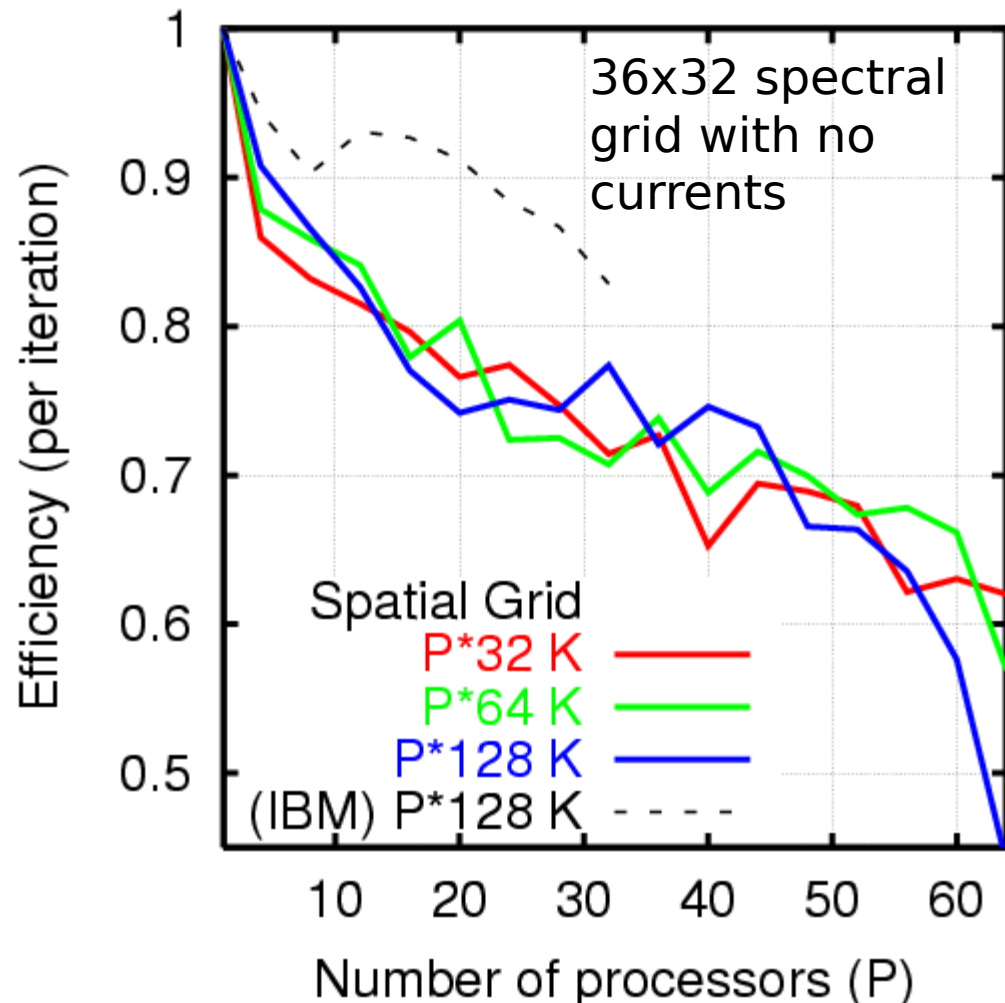
- IBM Regatta with 32 1.3GHz Power4
- Spatial grid scaled linearly with number of processors
- P ≤ min(X,Y)/16
- Total memory:
  – P*32 K: P*150 MB
  – P*64 K: P*300 MB
  – P*128 K: P*600 MB

Memory Constrained Scaling - SGI

- SGI O3K with 512 400MHz MIPS R12K
- Spatial grid scaled linearly with number of processors
- P ≤ min(X,Y)/16
- Total memory:
  - P*32 K: P*150 MB
  - P*64 K: P*300 MB
  - P*128 K: P*600 MB
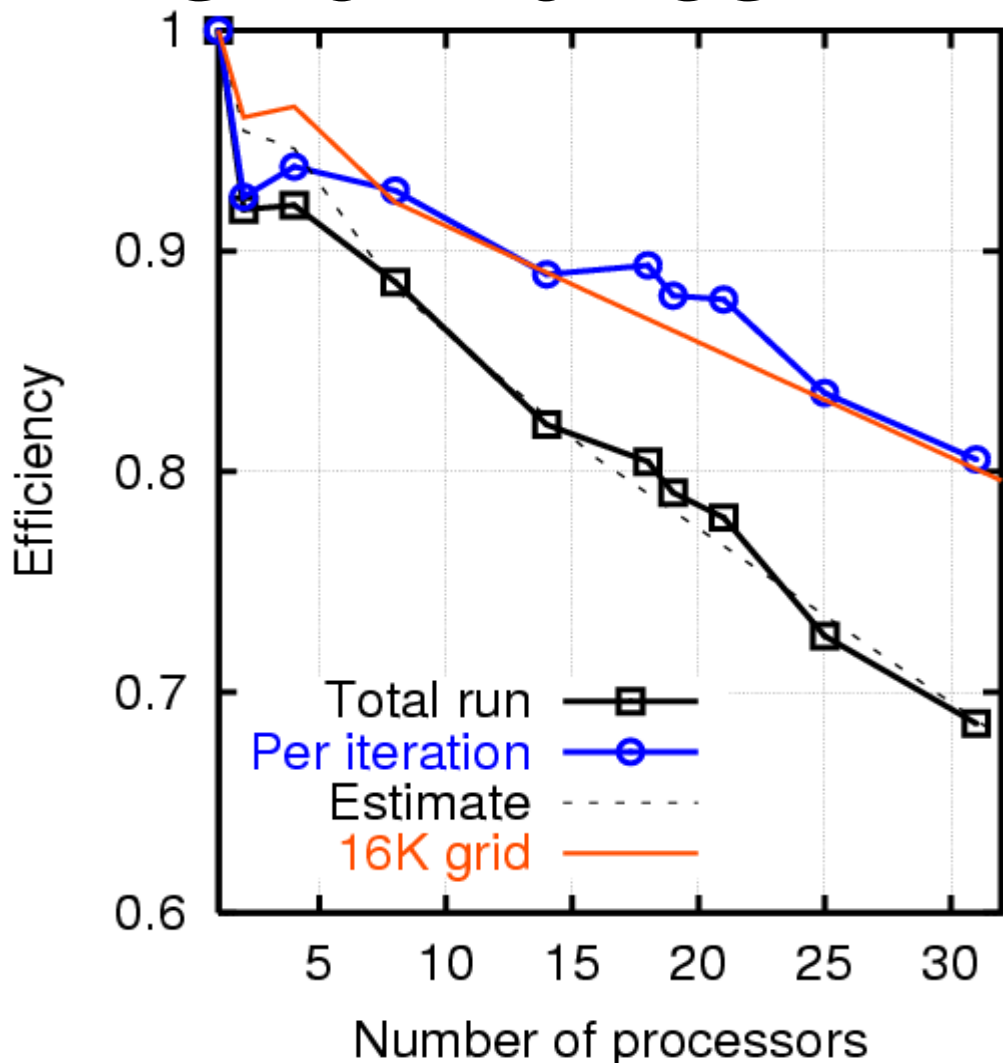
Lake Michigan Case

color scale : wave height (m)

contours : 0, 25, 100, 200m depth
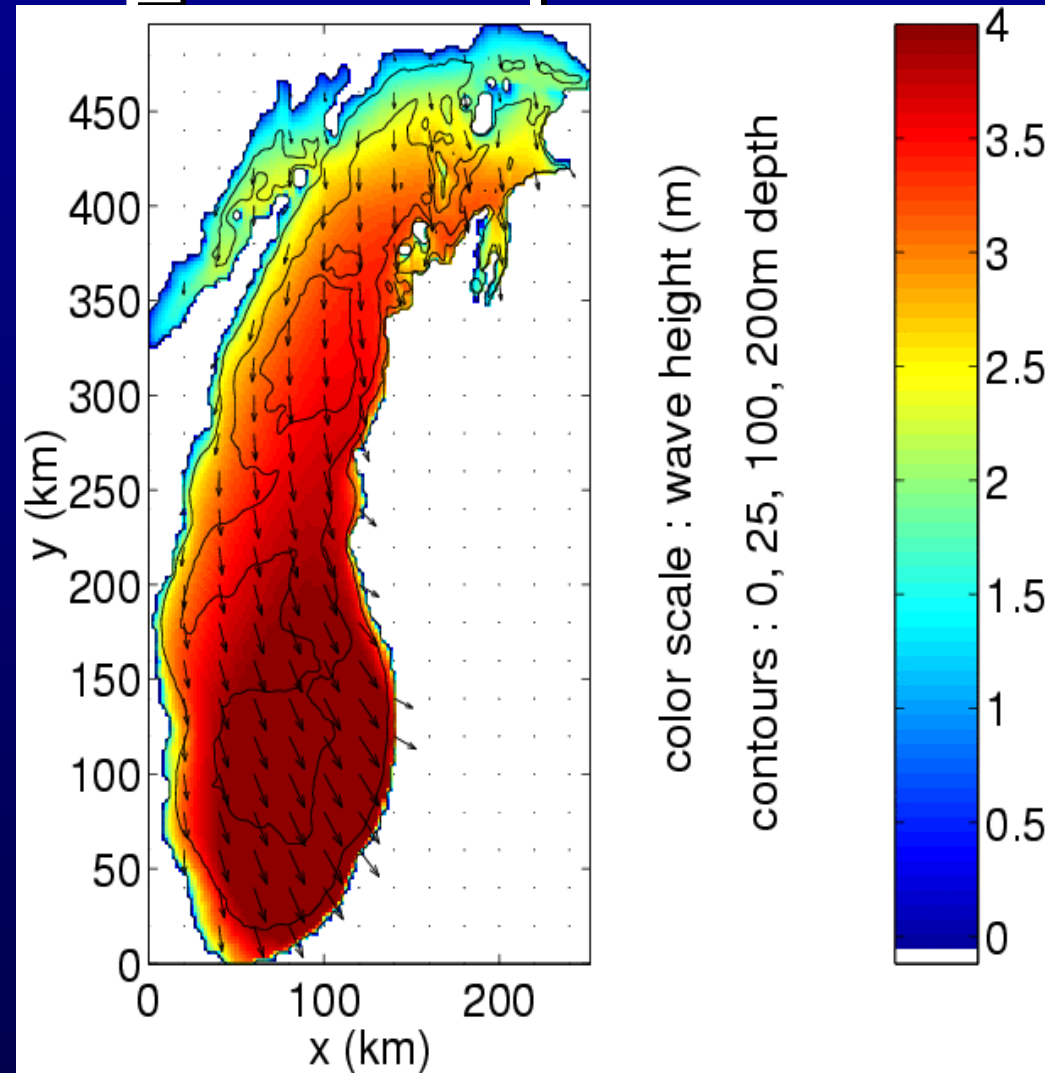
- Non-stationary, 4 day
- Time step = 0.1hr
- Hourly wind input
- 1hr & 2hr file output
- 2km spatial resolution
  - 126 x 248 mesh (31K)
  - 44% wet (13K)
- 36 x 32 spectral mesh
- No currents

Lake Michigan Parallel Performance

(IBM Regatta 32 x 1.3GHz Power4)

- 14.5 hr --> 1.0 hr on 18 processors
- Per iteration efficiency agrees with results for grid size close to number of wet points (16K)
- Total efficiency lower due to serial overhead from time-stepping and file I/O (~1% of single processor time)

Southern California Bight

Southern California Bight

contours : 0, 25, 100, 300m depth
color scale : wave heights (m)

Los Angeles
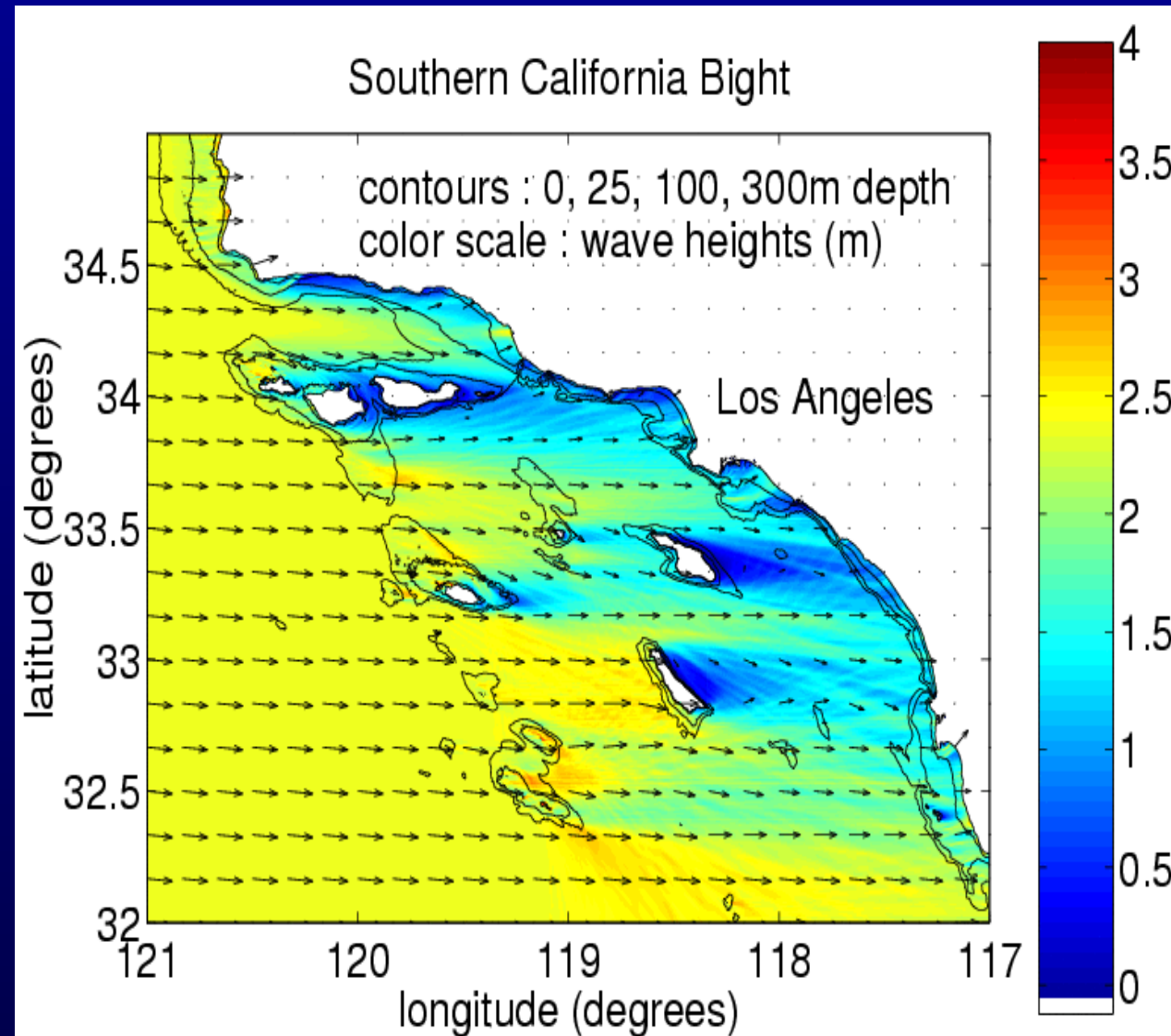
- Stationary case
- 2400 x 1800 spatial (66% wet)
- 36 x 11 spectral
- Includes full I/O
- 22 min on 30 processors of IBM Regatta
- Estimate serial time 8 hr (70 - 75% efficiency)

Summary

- A shared memory pipelined parallel version of SWAN has been implemented using OpenMP

- Parallel version is "bit-for-bit" compatible with the original serial version

- No change to user interface

- Modifications have been accepted for next official release of SWAN

- Parallel version will be transitioned into operational use at NAVOCEANO